METHOD AND APPARATUS FOR TESTING MICROARCHITECTURAL FEATURES BY USING TESTS WRITTEN IN MICROCODE

Cross Reference To Related Application(s)

This application is a continuation of application serial number 09/496,367, filed February 2, 2003, entitled "Method and Apparatus for Testing Microarchitectural Features By Using Tests Written In Microcode," now U.S. Patent No. _____, which is hereby incorporated by reference.

Technical Field

The technical field is related to mechanisms and methods for testing computer microarchitectures.

Background

The preferred method of emulating an instruction set on a microprocessor is to convert each emulated instruction (macroinstructions) into a series of instructions in the native instruction set (microinstructions). These microinstruction sequences are stored in microcode storage. In addition, the microprocessor may provide microinstructions that are only available for use by emulation hardware and not to code running in the native mode.

With traditional techniques for testing the emulated instruction set, test writers prepare sequences of user visible macroinstructions. The emulation hardware translates the macroinstructions into microinstructions that are then executed. In order to test certain microarchitectural features, the test writer must determine sequences of macroinstructions required to produce a desired sequence of microinstructions. The microinstruction sequences may be difficult (or even impossible) to construct with only macroinstructions. Further, a long sequence of macroinstructions may be required in order to produce the desired operands or machine state, leading to excessively long tests.

Summary

What is disclosed is an apparatus for testing a computer microarchitecture. The apparatus includes means for providing reprogrammed microcode. The means for providing the reprogrammed microcode includes means for reprogramming microcode, and means for storing reprogrammed microcode. The storing means includes microcode related to one or more macroinstructions, and reprogrammed test microcode for testing the computer microarchitecture, where the reprogrammed test microcode includes a sequence of microinstructions executed to test the computer microarchitecture. Finally, the apparatus includes means for sequencing the sequence of microinstructions and producing an address for the reprogrammed test microcode.

Also disclosed is a method for testing a computer microarchitecture, including the steps of reprogramming microcode for storage in a microcode storage, designating a

macroinstruction for execution, where the execution initiates a test sequence comprising the reprogrammed microcode, receiving inputs corresponding to entry points and computer state information, and producing an address for the reprogrammed microcode.

Finally, what is disclosed is a computer readable medium having code for conducting a test of a computer microarchitecture. The code implements the steps of mapping a macroinstruction to a particular sequence of microinstructions, replacing the particular sequence of microinstructions with an arbitrary set of microinstructions, wherein the arbitrary set of microinstructions comprises the test, receiving inputs corresponding to one or more of entry points and computer state information, and producing an address for the arbitrary set of microinstructions.

Brief Description Of The Drawings

The apparatus and method for testing microarchitectural features will be discussed in detail with reference to the following figures, wherein like numerals refer to like features, and wherein:

Figure 1 is a block diagram of an embodiment of an apparatus used for testing microarchitectural features;

Figure 2 is a logical diagram of a reprogrammable microcode storage; and

Figure 3 is a logical diagram of the conversion process between macroinstructions and microinstructions.

Detailed Description

Complex instruction set computer (CISC) architectures can be very powerful in that the CISC architecture allows for complicated and flexible ways of calculating elements such as memory addresses. The CISC instructions, or macroinstructions, may include one or more instructions of microcode. During development of a microprocessor, a test writer must verify that the macroinstructions executed by the microprocessor achieve the desired result. However, because the macroinstructions may include a number of microinstructions, the test writer may not be able to test the effect of each of the microcode instructions or a particular sequence of microinstructions by simply writing a test using the macroinstructions.

The process of writing a proper test sequence and verifying the function of a microprocessor design may be complicated when the microprocessor is designed to execute more than one instruction set. For example, a microprocessor may be designed to execute both CISC (e.g., IA-32) and reduced instruction set computer (RISC) (e.g., IA-64) instructions. In this example, the test writer must ensure that the CISC instructions are correctly emulated with desired sequences of microinstructions.

Furthermore, the test writer must reverse-engineer the sequence of macroinstructions that gives the desired sequence of microinstructions that is needed to test the microprocessor. This can be very difficult and time-consuming to accomplish. If

the designer then changes the microcode in the processor, the test writer may have to revise the test to recapture the original intended behavior. It is also possible that the behaviors that were originally possible in the microprocessor are made impossible by microcode changes or vise versa. By writing tests directly in microcode, the test writer solves both these problems.

In addition, during the early design stages of a microprocessor, the designer may be constantly revising the microcode used to emulate the macroinstructions. In doing this, the designer can introduce new sequences of microcode that have never been executed by the microprocessor. This can lead to latent bugs being discovered. By using tests written in microcode, sequences of microcode that are impossible in the normal operation of the microprocessor may be tested. This allows the designers to create a more robust design that will tolerate changes in the microcode sequences used to emulate the macroinstructions.

Finally, other hardware control structures on the chip can change during the design stages of a microprocessor. The designer must then determine if the changes will introduce errors in the microprocessor's behavior. This task may be eased if the designer can have the microprocessor design tested quickly and accurately. Use of microinstruction sequences can be an efficient way to test the new microprocessor designs.

Furthermore, during the early design stages of a microprocessor, the designer may be constantly revising the basic microcode operating on the microprocessor. The designer must then determine if the changes will introduce errors in the microprocessor's behavior. This task may be eased if the designer can have the microprocessor design tested quickly and accurately. Use of microinstruction sequences can be an effective way to test the new microprocessor designs.

Figure 1 is an overall block diagram of an apparatus for testing a computer microarchitecture using tests written in microcode. A test apparatus 10 includes a macroinstruction to microinstruction mapper 20. The macroinstruction to microinstruction mapper 20 receives one or more macroinstructions 11 and provides an output 31 to a microinstruction sequencer 30. The microinstruction sequencer 30 receives inputs corresponding to entry points for different events 33 and state information 32 from the computer microarchitecture or microprocessor to be tested. The microinstruction sequencer 30 outputs a microcode address sequence 35 to a microcode storage 40, some or all of which is reprogrammable by means known to those skilled in the art.

The microinstruction sequencer 30 may be implemented as a state machine to control operation of the reprogrammable microcode storage 40, and thereby control the sequence of microinstructions that are being executed.

The reprogrammable microcode storage 40 stores microcode instructions that may be issued to emulate the particular macroinstruction 11 entered into the macroinstruction to microinstruction mapper 20. The reprogrammable microcode storage 40 may store a large number of microinstructions. The reprogrammable microcode storage 40 will output one or more microinstructions 41_i to a microinstruction dispatcher 50.

The microinstruction dispatcher 50 sends particular microinstructions 51_i to corresponding execution units 61_i for execution of the microinstructions. The execution units 61_i may be integer execution units, floating point execution units, and branch units, for example.

A microcode reprogrammer 70 is used to reprogram microcode in the reprogrammable microcode storage 40. The microcode reprogrammer 70 may be used to reprogram any microinstruction or microinstruction sequence. The reprogrammed microinstruction sequence, which may be any arbitrary sequence of microinstructions, then constitutes the test that is to be run on the computer microarchitecture.

The test may be started by issuing a macroinstruction to the macroinstruction to microinstruction mapper 20. The resulting output 35 from the microinstruction sequencer 30 is used in the reprogrammable microcode storage 40 to execute the reprogrammed microinstruction sequence.

Figure 2 is a logical diagram of the reprogrammable microcode storage 40. The reprogrammable microcode storage 40 is shown as containing microcode for a number of macroinstructions. For example, the reprogrammable microcode storage 40 includes a microcode sequence for an ADD macroinstruction 42, a microcode sequence for a DIVIDE macroinstruction 43, and a microcode sequence for a SUBTRACT macroinstruction 44. As noted before, the reprogrammable microcode storage 40 may contain many more of these microcode lines. Which particular microcode lines are read out of the reprogrammable microcode storage 40 may be determined by the output 35 of the microinstruction sequencer 30 shown in Figure 1.

Figure 3 is a logical diagram showing the relationship between the macroinstruction and its corresponding microinstructions. In Figure 3, one particular variant of an ADD macroinstruction 21 includes MEM (memory) and REG (register) operands. In this case, "memory" means both the operand 1 and destination, and "register" is a second operand or operand 2. The ADD macroinstruction 21 maps to one or more microinstructions. As shown in Figure 3, the microinstructions include an address generation microinstruction 22, a load operand from memory microinstruction 23, an ADD operand 2 to data loaded microinstruction 24 and a store results to memory microinstruction 25. Thus, the microinstructions 22-25 are actually executed to emulate the ADD macroinstruction 21. The ADD macroinstruction 21 is intended to add to the

data specified in the memory location, the data that is specified in the register, and write the data back to memory into the same memory location.

Returning to Figure 1, assuming a test has been written in terms of macroinstructions, such as the ADD macroinstruction 21, the macroinstruction is read into the macroinstruction to microinstruction mapper 20, which then produces an entry point to the microinstruction sequencer 30. Once the microinstruction sequencer 30 has identified a particular sequence of microinstructions to be executed, the information 35 is fed to the reprogrammable microcode storage 40 and the design test is executed through the microinstruction dispatcher 50 and the execution units 61_i.

As can be seen from Figure 3, a test writer may desire to test the microprocessor's response to an ADD macroinstruction by specifying the ADD macroinstruction be executed. Furthermore, the test writer may want to test microprocessor response to the microinstructions in a sequence other than that specified by the particular macroinstruction, may desire to test microprocessor response to a series of the same or similar microinstructions, or may desire to test the microprocessor following design changes or changes to the microcode used to emulate the macroinstruction. For example, the test writer may desire to test microprocessor response to ten address generation microinstructions in sequence. However, the test writer may not be able to accomplish these aims by using or specifying a particular macroinstruction or sequence of macroinstructions.

Using the apparatus 10 shown in Figure 1, the test writer can specify any sequence of microinstructions to be executed on the microprocessor. For example, if the test writer desires to test the microprocessor by specifying ten address generation microcode instructions executed in sequence, the test writer could use the ADD instruction 21 shown in Figure 3, stripping out microinstructions 23-25 and using only the address generation microinstruction 22 but repeating this operation ten times. The reprogrammable microcode storage 40 could then be reprogrammed with reprogramming hardware 70 with the ten address generation microinstructions 22 and sequenced through the microprocessor to be tested. The test writer would then be easily able to test the microprocessor as it functions in the case of having to execute ten address generation microinstructions in a row.

The ability to test a microprocessor or chip using the apparatus 10 shown in Figure 1 may be implemented as a model operating on a computer workstation. That is, the functions represented by the modules shown in Figure 1 may be implemented as a functional model that can be used to test the early and subsequent stages of the microprocessor or chip design. Specifically, the apparatus 10 can be seen to be a model of one or more hardware devices programmed with specific functionality. Thus, the apparatus 10 may be a model of a design of an electronic device, and need not be

associated with a physical device. The apparatus 10 may perform in the same manner when the apparatus 10 is replaced with any program.

However, the components shown in Figure 1 may also be included as discrete hardware devices on a microprocessor or chip. In this case, the reprogrammable microcode storage 40 may require reprogramming in order to handle changes in test designs.

Furthermore, in the case where the apparatus 10 is a firmware or hardware program embodied on a physical device, the connections to the macroinstruction sources and other data sources can be hardware or software connections, as appropriate. Along the same lines of generality, the other components illustrated in Figure 1 can be firmware or hardware modules, rather than software modules.

If implemented as a software model, the modules or programs (both terms are used interchangeably) in Figure 1 can be stored or embodied on a computer readable medium in a variety of formats, such as source code or executable code, for example. Computer readable mediums include both storage devices and signals. Exemplary computer readable storage devices include conventional computer system RAM, ROM, EPROM (erasable, programmable ROM), EEPROM (electrically erasable, programmable ROM), and magnetic or optical disks or tapes. Exemplary computer readable signals, whether modulated using a carrier or not, are signals that a computer system hosting or running the apparatus 10 can be configured to access, including signals downloaded through the Internet or other networks.

The terms and descriptions used herein are set forth by way of illustration only and are not meant as limitations. Those skilled in the art will recognize that many variations are possible within the spirit and scope of the following claims, and their equivalents, in which all terms are to be understood in their broadest possible sense unless otherwise indicated.